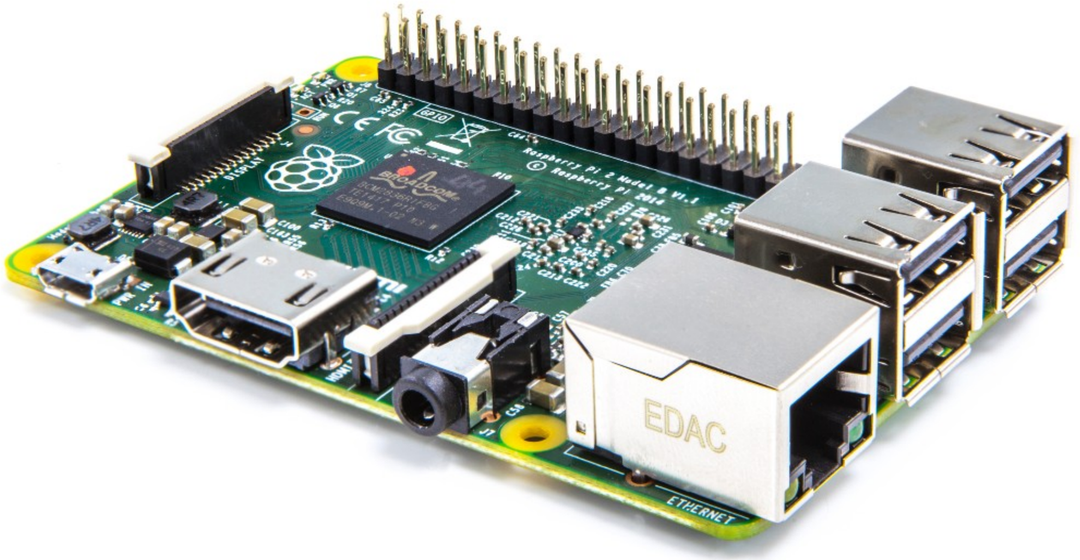
Pràctiques de control electrònic amb la interfície GPIO de

Raspberry Pi



(c) Artur Guillamet Sabaté
v1.0 (Maig, 2016)

Pràctiques de control electrònic amb la interfície GPIO de Raspberry Pi

INDEX Pàg.

1.- Introducció. 3
2.- Raspberry Pi. 4
3.- GPIO. 6
4.- Sistema Operatiu 8
5.- Programació bàsica GPIO. 9
6.- Pràctiques. 12
6.1- Pràctica 1. Control d'un Led. 12
6.2- Pràctica 2.- Control display de 7 segments. 14
7.- Exercicis. 19

1.- Introducció.

En aquest dossier es fa una introducció inicial al control de sistemes electrònics mitjançant la utilització de Raspberry Pi i la seva interfície GPIO.

Actualment s'han desenvolupat múltiples dispositius electrònics que per les seves capacitats i dimensions estan inserides dins d'aparells i coses quotidianes, subministrant-les-hi un nivell de "intel·ligència computacional" amb l'objectiu d'ajudar a les persones:

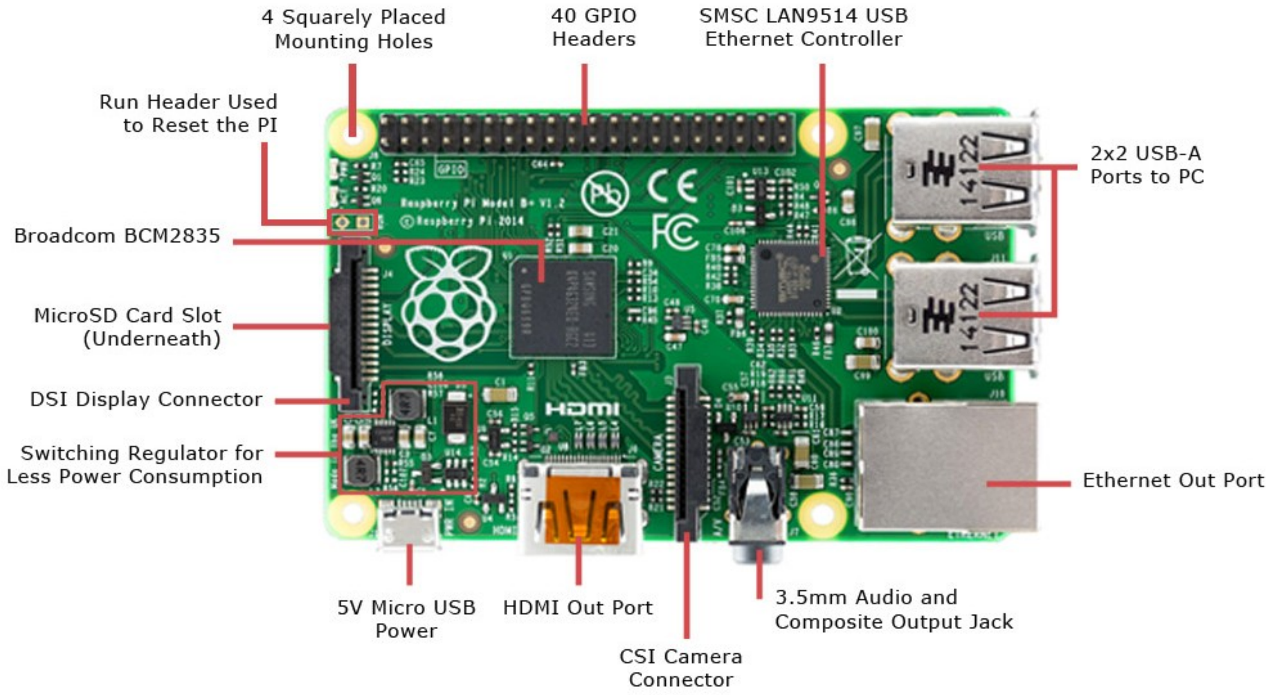
- Cotxes que prenen decisions per a la conducció o estacionament.
- Refrigeradors que gestionen l'stock de productes, caducitat, temperatura, etc. que tenen en el seu interior, i si de cas, en fan la comanda al supermercat.
- Habitatges que gestionen la seva eficiència energètica i confort òptims segons les preferències dels seus habitants.
- Impressores que detecten el nivell de càrrega del seu tòner i en fan la petició del recanvi automàticament al proveïdor
- etc.

Aquests dispositius que comuniquen, controlen, gestionen i ajuden a prendre decisions a les persones, són els que aporten nous avenços en el relativament incipient concepte de l'Internet de les coses (IoT): la xarxa d'objectes de la vida quotidiana interconnectats. Són els objectes, les coses, les que es connecten a Internet per conèixer informació de altres coses que també estan connectades a Internet i es transfereixen informació entre elles i amb les persones.

Amb l'estudi de Raspberry Pi, es pretén donar una introducció tècnica amb un dispositiu que pot afegir unes possibilitats d'interacció a les coses quotidianes del nostre entorn.

2.- Raspberry Pi.

Raspberry Pi és un ordinador monoplaca de baix cost desenvolupat per la Fundació Raspberry Pi del Regne Unit amb l'objectiu d'estimular l'ensenyament de la Informàtica.



Principals elements que formen l'ordinador monoplaca Raspberry Pi 2 model B.

Existeixen actualment múltiples ordinadors monoplaca en el mercat, però, no tots ells disposen d'una interfície de propòsit general per connectar directament amb circuits elèctrics i electrònics com és la interfície GPIO. Això és la principal qualitat que ens fa interessant aquesta placa per les aplicacions que li volem implementar.

Podem trobar diversos models, amb més o menys prestacions i cost i diferents evolucions temporals. En aquest dossier es prioritza la descripció del model: Raspberry Pi 2 B.

Nucli:
El model 2 B esta basat en el xip SOC (System on Chip) Broadcom BCM 2836 que incorpora la CPU (ARM V7 4 cores a 900MHz, el GPU processador de video, 1Gby de SDRAM i DSP processador digital de senyals).

Audio / Vídeo:
A la mateixa placa ens trobem una sortida d'àudio i vídeo mitjançant el connector HDMI a 1080p per poder-la connectar a televisors i monitors actuals que, habitualment ja tenen aquesta connexió. Simultàniament, també disposa d'un connector mini jack que permet la connexió dels dos canals d'àudio i el senyal de vídeo compost, útil per connectar la placa a dispositius televisors més antics que solament disposen d'aquestes entrades d'àudio i vídeo amb connectors RCA o Scart.

Comunicacions:
Incorporat dins de la mateixa placa disposem d'un connector RJ45 que ens proveeix d'una connexió Ethernet a 10/100 Mbps i 4 ports USB. El model 3 també inclou incorporat a la placa de la interfície Wifi i Bluetooth.

Emmagatzematge:
Per ubicar el sistema operatiu i l'espai de dades de l'usuari, la placa disposa d'un connector per a targetes micro SD, que, amb un mínim de 4GB s'hi pot instal·lar el sistema operatiu dels que es troben disponibles a la web oficial de Raspberry Pi. Com més gran sigui aquest microSD, més espai hi haurà disponible per a les dades d'usuari.

	Quadre resum de característiques tècniques per al model: Raspberry Pi 2 B
SoC	Broadcom BCM2836
CPU	ARM11 ARMv7 Cortex-A7 4nuclis a 900MHz
GPU	Broadcom videocore IV 250MHz OpenGL ES 2.0
RAM	1 Gb SDRAM 450MHz
Ports USB	4
GPIO	40 pins
Vídeo	HDMI 1.4 1920x1200
Emmagatzematge	microSD
Ethernet	1 x 10/100Mbps
Mida	85,6 x 56,5 mm
Pes	45 g.

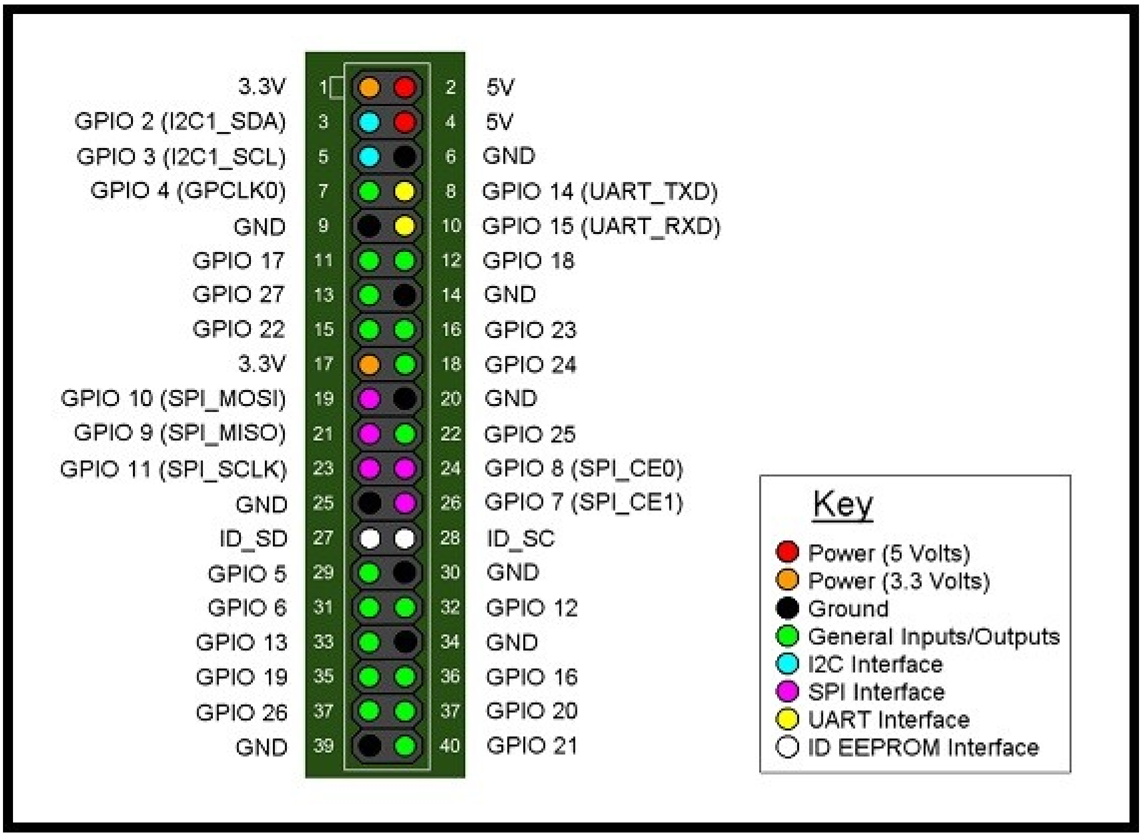
Tot i que la tarja s'alimenta mitjançant un connector micro USB on obté una tensió de 5 v, la circuiteria electrònica interna funciona, també, a tensions més baixes: 3'3 v o inferior, 2'5v i 1'8v.

Per això, dins la placa s'hi troba un regulador commutat de tensió que converteix la tensió de 5 v subministrada externament als 3'3v, 2'5v i 1'8v de treball dels diferents components interns de la placa. En aquest aspecte, cal tenir en compte que els pins de la interfície GPIO treballen a 3'3v.

3.- GPIO

La interfície GPIO (General Purpose Input/Output) és un port d'expansió format per un conjunt de pins que permetran interconnectar la Raspberry Pi amb els dispositius electrònics perifèrics que vulguem controlar.

En els models 2 i 3 de Raspberry Pi està format per 40 pins que es distribueixen en les següents funcions:



Distribució dels pins i les seves funcions a la interfície GPIO

En l'esquema de la distribució dels pins GPIO, s'observa com no hi ha una correspondència amb el número de pin i el número corresponent al GPIO, en la programació caldrà determinar si fem referència a un valor o l'altre.

Els pins GPIO permetran a la Raspberry Pi comunicar-se amb l'exterior tan per activar elements com per llegir el seu estat. La tensió de treball dels pins és de 3'3 v. Per tant, un "1" lògic correspondrà a una tensió de 3'3v i un "0" lògic correspondrà a una tensió de 0v. La màxima corrent que es pot subministrar és de 16 mA. Es important no confondre aquests valors i evitar aplicar voltatges superiors als pins com a entrada, o sobrecorrents, ja que es possible provocar una avaria als xips interns per sobretensió (concretament els pins es corresponen amb els seus equivalents en el xip SoC que seria l'afectat per sobretensió o sobrecorrent). Per això cal vigilar el funcionament de determinats sensors que funcionen a 5v i subministren una tensió de sortida d'aquest valor que pot afectar als pins GPIO.

Tots els pins GPIO es poden gestionar directament mitjançant codi, ja sigui per posar-hi un valor o llegir-lo.

A més, alguns dels pins poden tenir múltiples funcionalitats addicionals a la de I/O digital. Algunes funcions més complexes, que són gestionades pel mateix xip són:

Universal Asynchronous Receiver/Transmitter (UART):

UART és un bus de connexió sèrie asíncron quasi-compatible amb l'estàndard RS-232 (la diferència està en que RS-232 treballa a 12v i els connectors del GPIO treballen a 3'3v). Serveix per comunicar, via connexió sèrie, la Raspberry Pi i, únicament, un altre dispositiu sèrie, simultàniament. Els pins afectats són:

- pin 8 TX Transmit
- pin 10 RX Receive

Inter-integrated circuit (I^2C):

I^2C és un bus de comunicacions sèrie entre circuits integrats, en el que s'hi poden connectar múltiples perifèrics I^2C esclaus i un master. La Raspberry Pi actua com a master. Cada perifèric I^2C és referenciat per un número i les comunicacions es transmeten a cada esclau per un únic parell de cables de comunicació (més el GND), així és possible tenir molts perifèrics connectats i una simple connexió amb solament dos cables. Aquests pins són:

- pin 3 SDA Data
- pin 5 SCL Clock.

Serial Peripheral Interface (SPI)

SPI és un bus de comunicacions sèrie síncron (amb rellotge, clock) on s'hi pot connectar múltiples perifèrics SPI esclaus. Raspberry Pi es configura com a mestre del bus. Els pins del bus SPI són:

- pin 19 MOSI Master Out, Slave In
- pin 21 MISO Master In, Slave Out
- pin 23 SCLK Serial Clock
- pin 24 CE0 Channel Enable 0. També conegut com Slave Select 0 (SS0)
- pin 26 CE1 Channel Enable 1. També conegut com Slave Select 1 (SS1)

4.- Sistema Operatiu

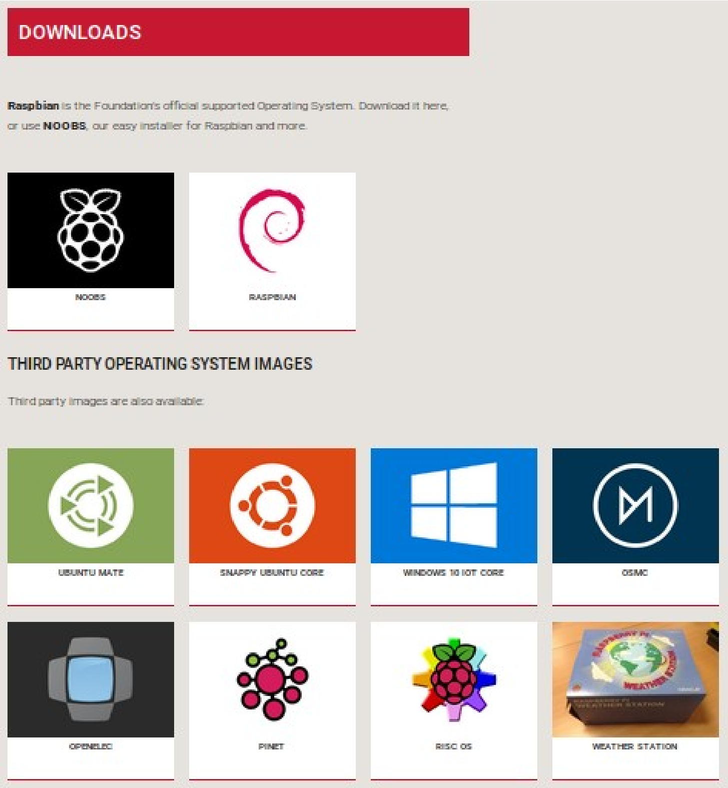
Com a tal sistema informàtic que és la Raspberry Pi, aquesta requereix d'un sistema operatiu per gestionar tots els seus dispositius interns, els perifèrics i facilitar la interacció amb les persones i amb altres dispositius.

La mateixa fundació Raspberry Pi ofereix diverses opcions de sistemes operatius adaptats a l'arquitectura ARM. Entre elles, la seva distribució oficial Linux anomenada Raspbian, i el carregador de distribucions NOOBS que permet tenir més d'una distribució i elegir la activa en l'engegada.

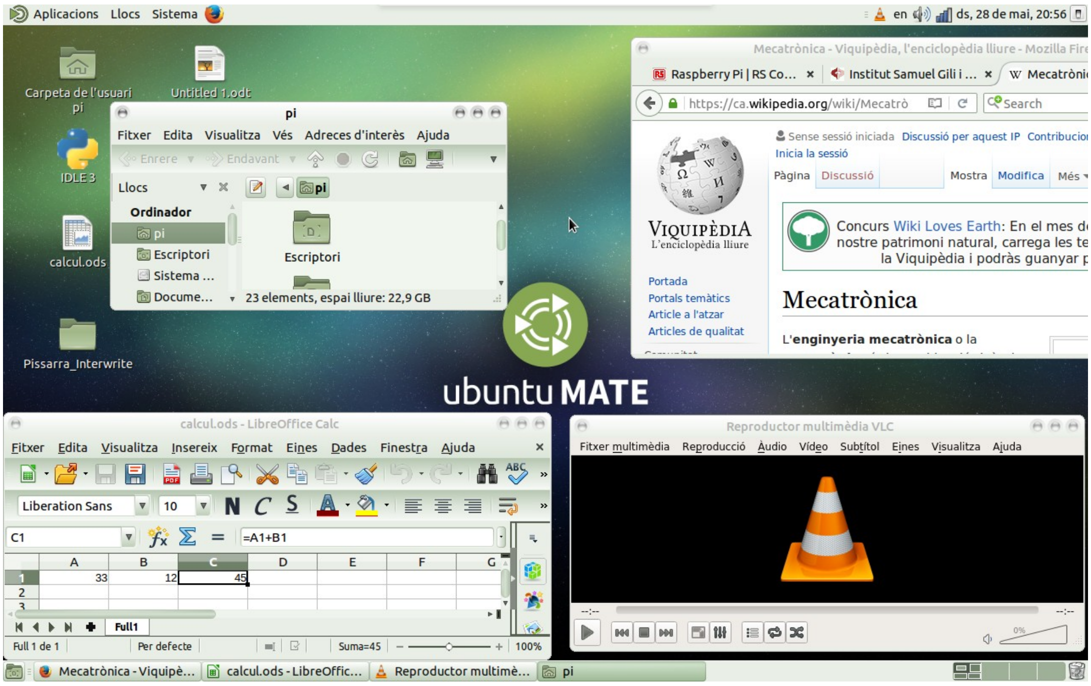
Però, també existeix altres distribucions Linux que han estat adaptades per a Raspberry Pi per part de altres entitats com ara la Ubuntu Mate, Openelec, OSMC, OpenSuse, etc.

El sistema operatiu es pot trobar i descarregar a la web oficial: https://www.raspberrypi.org/downloads/. Les diverses opcions són variants del SO Linux, encara que també és pot trobar alguna opció amb el SO Windows 10 IoT.

La instal·lació d'un o altre SO consisteix en descarregar-se la imatge corresponent i fer-ne una còpia a la tarja microSD seguint el procediment que ja s'explica en la mateixa pàgina web de descàrrega.



Diverses distribucions disponibles per descarregar des de la web oficial de Raspberry Pi



Mostra d'un escriptori de la distribució Ubuntu Mate sobre una Raspberry Pi

L'accessibilitat a aquest sistema operatiu pot ser la clàssica connectant el teclat, ratolí i pantalla, o bé , aprofitant les facilitats d'interconnexió remota pròpies de Linux.

5.- Programació bàsica GPIO. Python.

Els pins de la interfície GPIO són accessibles des del mateix sistema operatiu implementat a la mateixa Raspberry Pi. Això facilita la possibilitat de crear programes informàtics que en llegeixin el seu estat i en puguin modificar el valor.

L'accés a la interfície GPIO s'ha de fer amb permisos de superusuari, de cas contrari el SO retorna un error per manca de permisos. Aixo fa que qualsevol programa que manipuli els pins GPIO s'ha d'executar com a root (o equivalent).

Gairebé tots els llenguatges de programació habituals en les distribucions Linux de la Raspberry Pi disposen, a més de les instruccions de llenguatge habitual, unes instruccions específiques per manipular els GPIO. En la majoria de llenguatges de programació, aquestes instruccions específiques s'han de carregar, al inici del programa, com a llibreries que en faciliten la programació, en altres casos com els llenguatges nadius, es pot fer la programació directament des del mateix llenguatge.

Entre altres llenguatges, es pot programar amb: C/C++, Java, PHP, BashScript, Perl, Python, Basic, Scratch, etc. Podem trobar informació sobre característiques específiques per programar en qualsevol dels llenguatges indicats a la web: http://elinux.org/RPi_GPIO_Code_Samples

Entre els llenguatges més habituals de programació per Raspberry Pi tenim el Python, instal·lat i disponible per defecte en la majoria de les distribucions de Raspberry Pi.
A continuació es descriu les principals característiques d'aquest llenguatge de programació interpretat i la seva aplicació expressa al control GPIO.

Python

Característiques del llenguatge Python

Python és un llenguatge de programació interpretat de propòsit general, amb llicència de codi obert, compatible amb la Llicència Pública GNU, de creació recent, la versió 0.9.0 es va alliberar el 1991 i la 1.0 el 1994 per part del seu creador, Guido van Rossum. Ha tingut una popularitat creixent entre els programadors per la seva senzillesa, versatilitat, adaptació a diferents entorns operatius i multiplataforma.

L'entorn de desenvolupament del llenguatge Python (IDLE, Integrated Development and Learning Environment) en permet l'execució interactiva d'instruccions des de la línia d'ordres o mitjançant scripts executables. És ràpid desenvolupar aplicacions en aquest entorn, encara que té les limitacions de velocitat d'execució, pròpies de tots els llenguatges interpretats.

Pretén ser un llenguatge altament llegible, per tant, cada instrucció s'ha de trobar en una sola línia i els blocs de codi s'identifiquen mitjançant la **indentació** que és d'utilització obligatòria. Així, a diferència de altres llenguatges, com C, on cada bloc de codi s'ha de trobar entre claus { } i la indentació és opcional, per Python, solament la indentació iniciada per **:** defineix els diferents blocs de codi encabits dins de estructures tipus for, while, if, etc. o dins de les pròpies funcions i classes.

Admet gairebé tots els principals paradigmes de la programació de forma directa o mitjançant l'afegit de mòduls específics: programació imperativa, programació estructurada, programació orientada a objectes, programació funcional, etc.

Les **instruccions** s'han de trobar en una sola línia i solament hi ha d'haver una instrucció. Només el salt de línia fa de separador de les diferents instruccions. Si alguna instrucció és vol distribuir en dos o més línies, cal encadenar les línies posant un barra invertida: \ al final de línia per indicar que continua a la següent.

Els **comentaris** de una sola línia són els que comencen amb el símbol: **#** , si volem posar comentaris de múltiples línies, llavors utilitzarem els tres apòstrofs seguits: **'''** a l'inici i final del comentari.

Les **variables** es defineixen dinàmicament, per tant, no fa falta declarar-les prèviament ni tampoc indicar el seu tipus. Dinàmicament s'estableix el tipus de dades que li correspon segons la dada que se li assigna en el primer moment, o bé, posteriorment, ja que, en l'evolució del programa pot canviar de tipus de dades en cas que se li assigni altres tipus.

Les **operacions lògiques** s'indiquen amb les paraules clau: not, and i or.

L'**operador d'assignació** és = i els **operadors de comparació** són: ==, !=, >, <, >=, <=

Les **instruccions condicionals** bàsiques són: if, elif, else

Les **instruccions iteratives** bàsiques són: for i while

Les **funcions** és defineixen amb: *def nomfuncio(paràmetres)* i, solament en cas de retornar un valor, contindrà la sentència *return.*

Les **classes** es defineixen amb *class*.

Programació de la interfície GPIO

Si volem programar la interfície GPIO des d'un programa Python, cal carregar la llibreria GPIO, que ja està incorporada en el mateix SO de la Raspberry Pi. A més, el programa executable ha de tenir permisos de root o similar, de cas contrari, donarà un error en intentar accedir al control de la GPIO per falta de permisos.

La forma de carregar la llibreria i les primeres configuracions és la següent:

`import Rpi.GPIO as GPIO`	Càrrega de la llibreria GPIO
`GPIO.setwarnings(False)`	En cas que no vulguem veure els avisos no crítics
`GPIO.setmode(GPIO.BOARD)`	Referenciarem els pins segons el seu número de pin, alternativament podríem posar GPIO.BCM, llavors referenciaríem els pins segons el seu número de GPIO.
`GPIO.setup(pin, tipus)`	Establim que un determinat pin sigui de entrada o sortida. pin – és el número de pin. tipus – val *input* o *output* segons vulguem que sigui entrada o sortida.
`GPIO.input(pin)`	Llegim l'estat lògic del número de pin indicat a *pin*.
`GPIO.output(pin, estat)`	Establim el nivell lògic (1 o 0) indicat per *estat* al pin indicat per *pin*.

Així, un programa Python que interactua amb la interfície GPIO haurà de començar amb algunes línies com les següents o similars en cas d'establir altres condicions inicials:

```
import Rpi.GPIO as GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
```

i múltiples línies `GPIO.setup(...)` per indicar el sentit dels pins utilitzats.

Ja dins el programa, diverses línies amb `GPIO.input(...)` i `GPIO.output(...)` per llegir o establir, respectivament, el valor lògic dels pins indicats.

6.- Pràctiques.

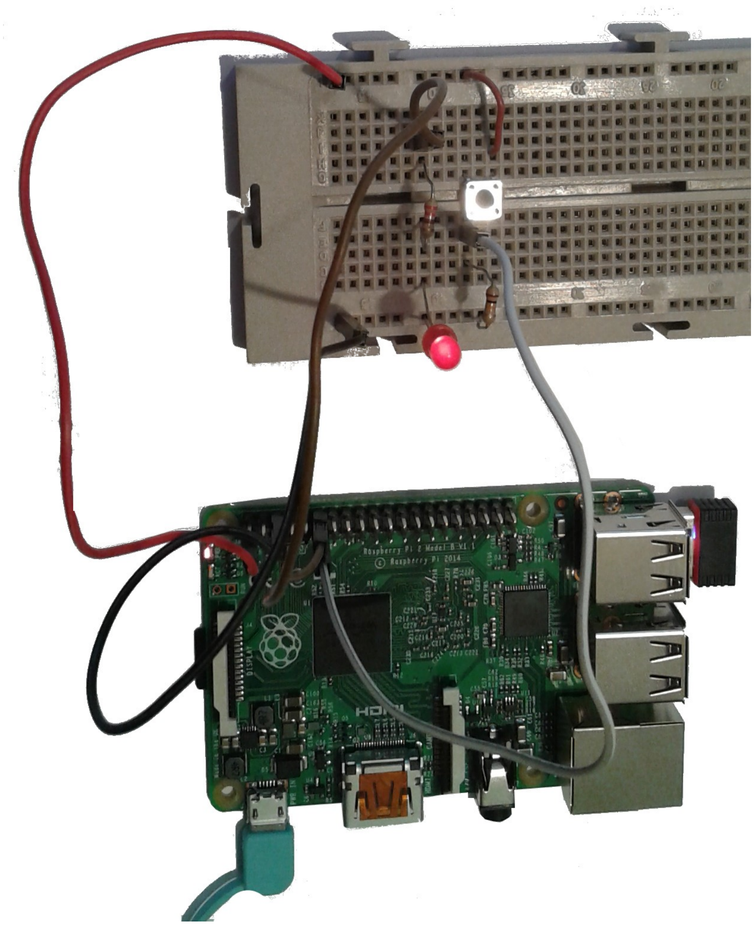
6.1.- Pràctica 1. Control d'un Led.

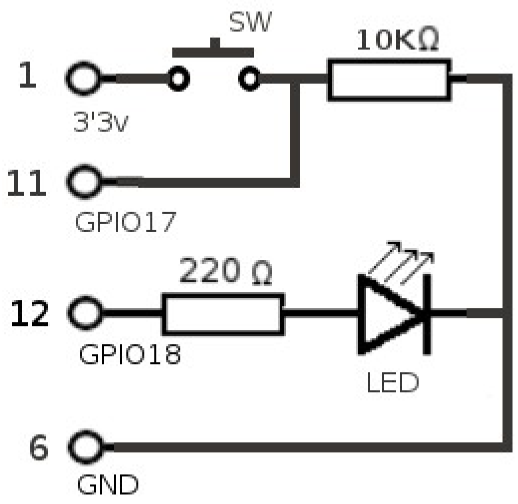
Tenim un led connectat en el pin 12 i un polsador connectat al pin 11 de la interfície GPIO de Raspberry Pi. Volem que el led parpellegi amb una cadència de 0'8 segons encès i 0'8 segons apagat, però, quan es premi el polsador, la cadència de parpelleig augmenti a 0'1 segons encès i 0'1 segons apagat. Al mateix temps volem que ens mostri per la pantalla l'estat de l'entrada del polsador i si el led està encès o apagat.

Components: 1 polsador NO, 1 resistència de 10KΩ, un led, 1 resistència de 220Ω.

Per fer aquesta pràctica inicial, el circuit a connectar als pins de la interfície GPIO és el següent:

Esquema:

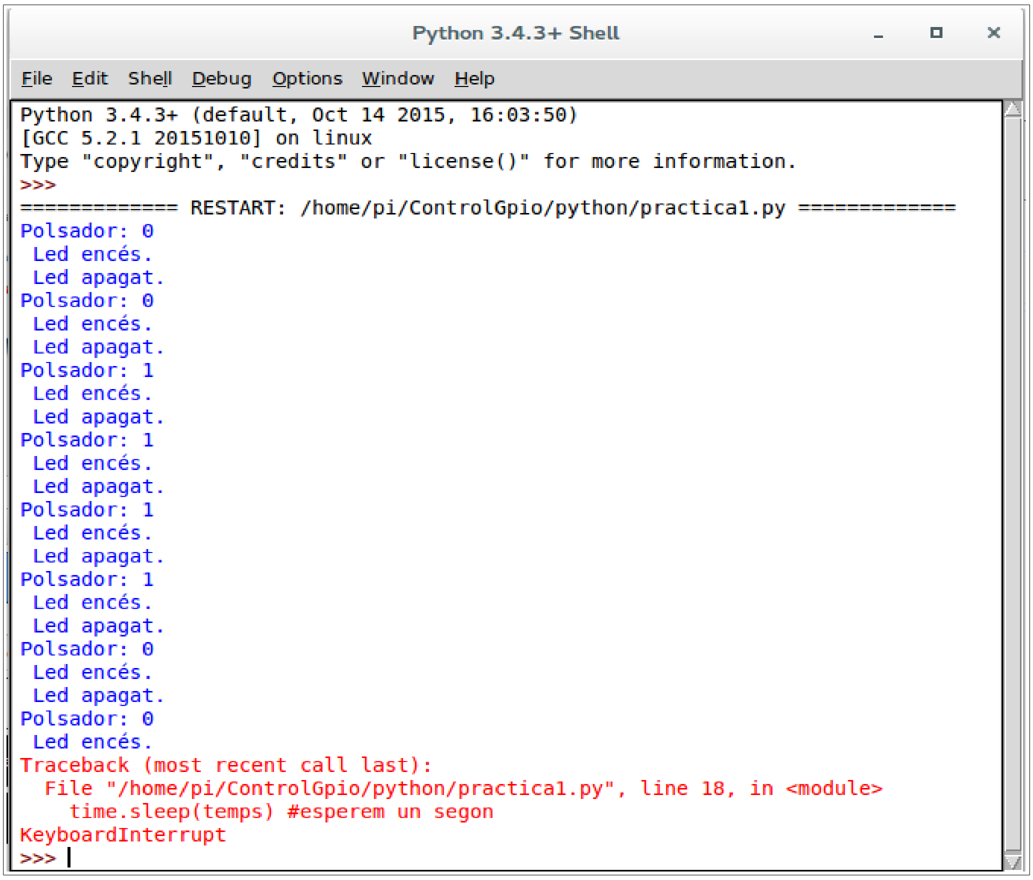




Programa:

practica1.py

```
import RPi.GPIO as GPIO  #importem la llibreria gpio
import time #importem la llibreria time
GPIO.setwarnings(False) #evitem que es mostrin els warnings.
GPIO.setmode(GPIO.BOARD) #referenciarem els pins segons el seu número de pin.
GPIO.setup(12, GPIO.OUT) #definim el pin 12 com a sortida
GPIO.setup(11, GPIO.IN) #definim el pin 11 com a entrada
#led al pin 12 que parpalleja
while True: #bucle infinit que aturarem amb Cntl-C
    valPolsador=GPIO.input(11)
    if valPolsador == 1:
        temps=0.1
    else:
        temps=0.8
    print("Polsador: " + str(valPolsador))
    GPIO.output(12, 1) #l'activem a 1
    print(" Led encés.")
    time.sleep(temps) #esperem un segon
    GPIO.output(12, 0) #l'apaguem a 0
    print(" Led apagat.")
    time.sleep(temps) #esperem un segon
```




```
Python 3.4.3+ (default, Oct 14 2015, 16:03:50)
[GCC 5.2.1 20151010] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
============= RESTART: /home/pi/ControlGpio/python/practica1.py =============
Polsador: 0
 Led encés.
 Led apagat.
Polsador: 0
 Led encés.
 Led apagat.
Polsador: 1
 Led encés.
 Led apagat.
Polsador: 1
 Led encés.
 Led apagat.
Polsador: 1
 Led encés.
 Led apagat.
Polsador: 1
 Led encés.
 Led apagat.
Polsador: 0
 Led encés.
 Led apagat.
Polsador: 0
 Led encés.
Traceback (most recent call last):
  File "/home/pi/ControlGpio/python/practica1.py", line 18, in <module>
    time.sleep(temps) #esperem un segon
KeyboardInterrupt
>>>
```

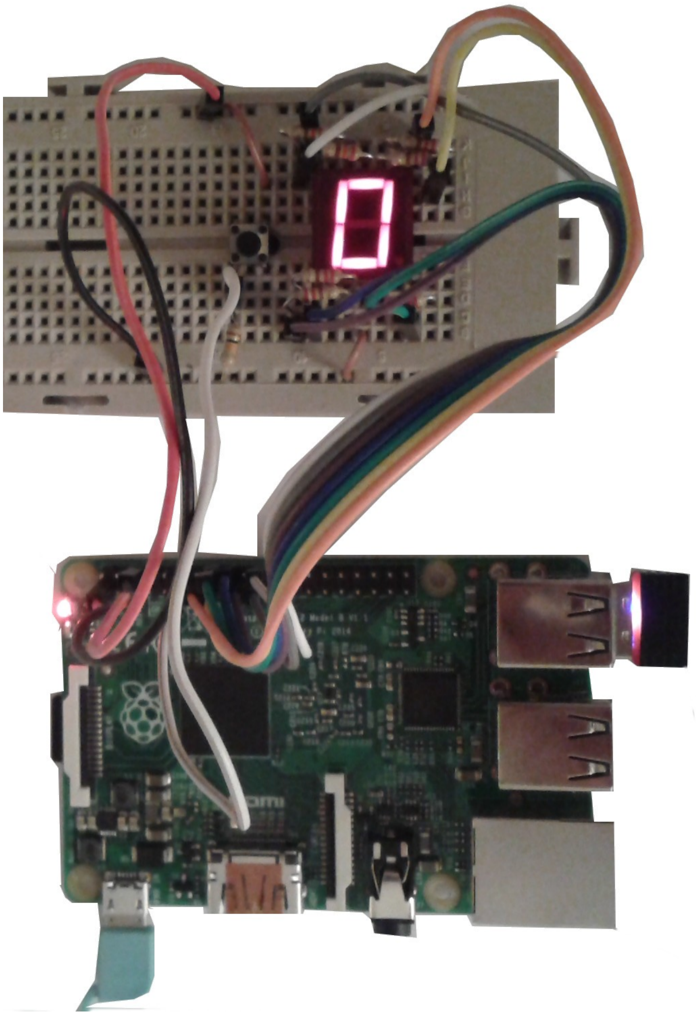
6.2.- Pràctica 2.- Control display de 7 segments.

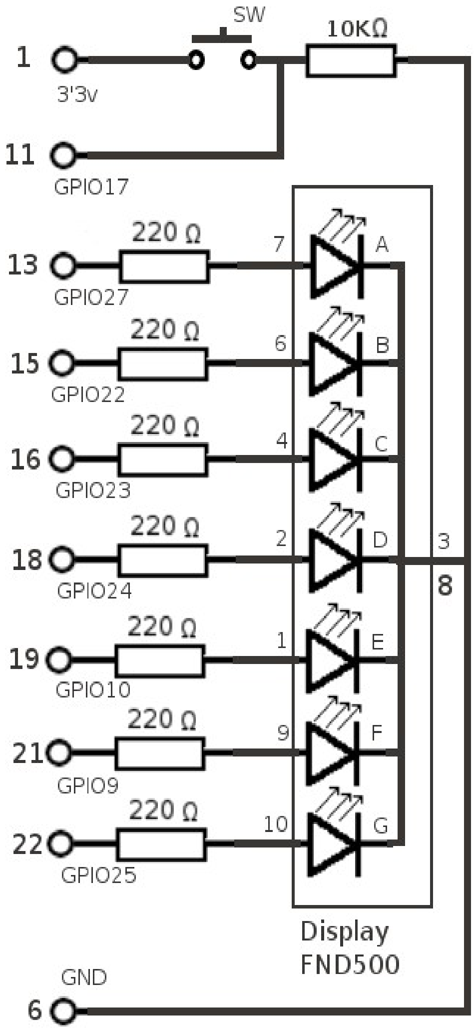
Volem utilitzar un display de 7 segments per que compti contínuament, de forma cíclica, entre 0 i 9 mentre s'està prement un polsador. En canvi, si deixem de prémer el polsador, el comptatge s'ha d'aturar i en el display s'ha de mostrar fix el darrer número comptat.

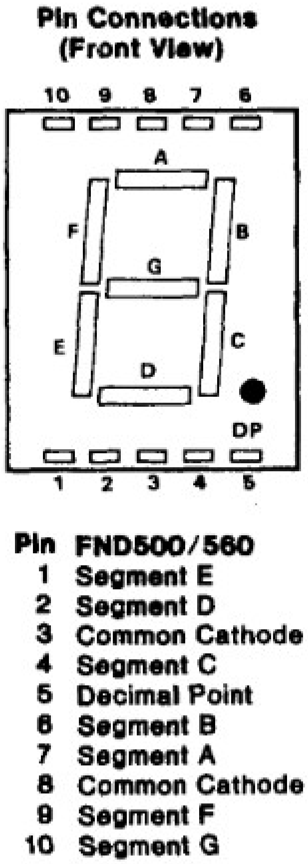
Components: 1 polsador NO, 1 resistència de 10KΩ, un display de 7 segments FND500 de càtode comú, 7 resistències de 220Ω.

Per muntar el circuit podem utilitzar un display de 7 segments tipus FND500 de càtode comú que consta dels 7 segments i un punt. Cada segment i el punt són un led que s'il·lumina en la forma rectangular que li correspon al segment. Els set segments s'anomenen amb les lletres A .... G. En el diagrama següent veiem quines potes i la seva numeració corresponen a cada segment d'aquest model de display.

Esquema:







Programa:

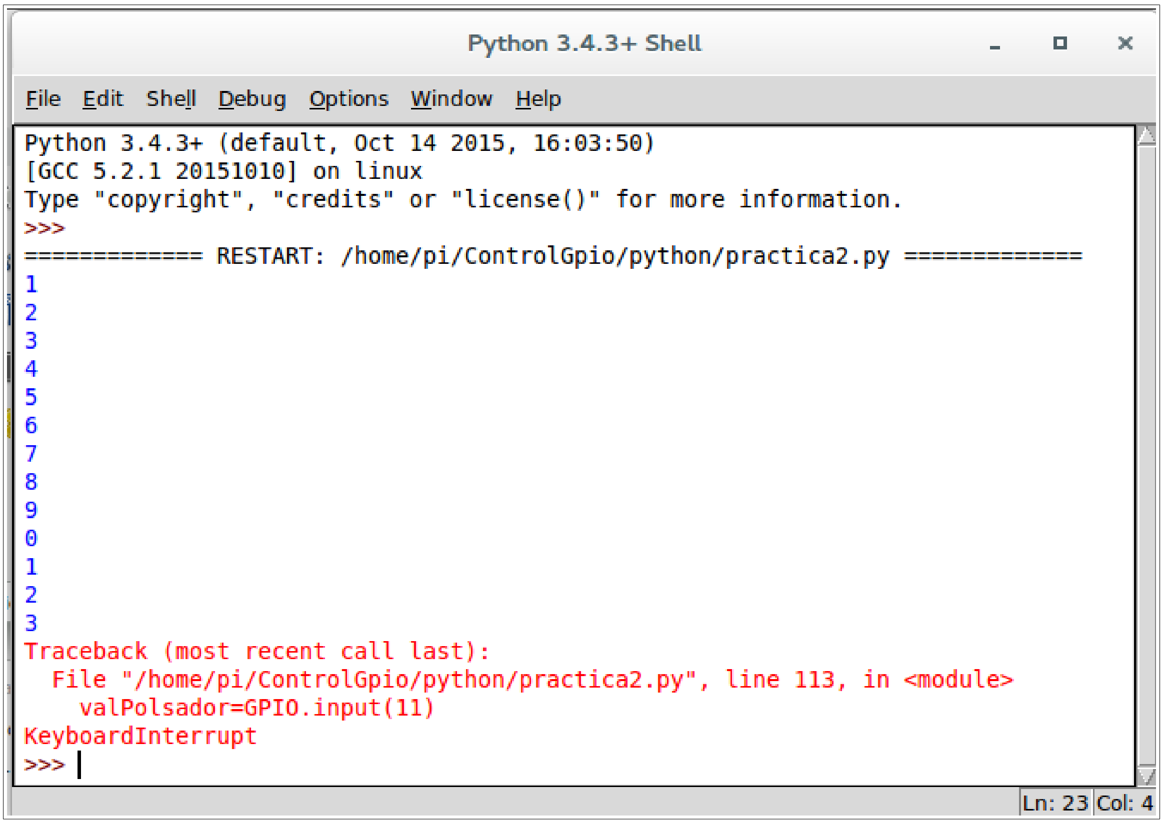
practica2.py

```
import RPi.GPIO as GPIO  #importem la llibreria gpio
import time #importem la llibreria time
GPIO.setwarnings(False) #evitem que es mostrin els warnings.
GPIO.setmode(GPIO.BOARD) #referenciarem els pins segons el seu número de pin.
GPIO.setup(11, GPIO.IN) #definim el pin 11 com a entrada
#definim els pins del display com a sortida:
GPIO.setup(13, GPIO.OUT) #a
GPIO.setup(15, GPIO.OUT) #b
GPIO.setup(16, GPIO.OUT) #c
GPIO.setup(18, GPIO.OUT) #d
GPIO.setup(19, GPIO.OUT) #e
GPIO.setup(21, GPIO.OUT) #f
GPIO.setup(22, GPIO.OUT) #g
```

```
#definim la funció display
#aquesta funció mostra el número que li indiquem en el display connectat
#li passem els paràmetres:
# num - número a visualitzar entre 0 i 9, si es un altre, esborra el display.
# a,b,c,d,e,f,g - pins on està connectat el segment corresponent.
def display(num,a,b,c,d,e,f,g):
    if num==0:
        GPIO.output(a,1)
        GPIO.output(b,1)
        GPIO.output(c,1)
        GPIO.output(d,1)
        GPIO.output(e,1)
        GPIO.output(f,1)
        GPIO.output(g,0)
    elif num==1:
        GPIO.output(a,0)
        GPIO.output(b,1)
        GPIO.output(c,1)
        GPIO.output(d,0)
        GPIO.output(e,0)
        GPIO.output(f,0)
        GPIO.output(g,0)
    elif num==2:
        GPIO.output(a,1)
        GPIO.output(b,1)
        GPIO.output(c,0)
        GPIO.output(d,1)
        GPIO.output(e,1)
        GPIO.output(f,0)
        GPIO.output(g,1)
    elif num==3:
        GPIO.output(a,1)
        GPIO.output(b,1)
        GPIO.output(c,1)
        GPIO.output(d,1)
        GPIO.output(e,0)
        GPIO.output(f,0)
        GPIO.output(g,1)
    elif num==4:
        GPIO.output(a,0)
        GPIO.output(b,1)
        GPIO.output(c,1)
        GPIO.output(d,0)
        GPIO.output(e,0)
        GPIO.output(f,1)
        GPIO.output(g,1)
    elif num==5:
        GPIO.output(a,1)
        GPIO.output(b,0)
        GPIO.output(c,1)
        GPIO.output(d,1)
        GPIO.output(e,0)
        GPIO.output(f,1)
        GPIO.output(g,1)
    elif num==6:
        GPIO.output(a,1)
        GPIO.output(b,0)
        GPIO.output(c,1)
        GPIO.output(d,1)
        GPIO.output(e,1)
        GPIO.output(f,1)
        GPIO.output(g,1)
```

```
    elif num==7:
        GPIO.output(a,1)
        GPIO.output(b,1)
        GPIO.output(c,1)
        GPIO.output(d,0)
        GPIO.output(e,0)
        GPIO.output(f,0)
        GPIO.output(g,0)
    elif num==8:
        GPIO.output(a,1)
        GPIO.output(b,1)
        GPIO.output(c,1)
        GPIO.output(d,1)
        GPIO.output(e,1)
        GPIO.output(f,1)
        GPIO.output(g,1)
    elif num==9:
        GPIO.output(a,1)
        GPIO.output(b,1)
        GPIO.output(c,1)
        GPIO.output(d,1)
        GPIO.output(e,0)
        GPIO.output(f,1)
        GPIO.output(g,1)
    else:
        GPIO.output(a,0)
        GPIO.output(b,0)
        GPIO.output(c,0)
        GPIO.output(d,0)
        GPIO.output(e,0)
        GPIO.output(f,0)
        GPIO.output(g,0)


n=0
while True: #bucle infinit que aturarem amb Cntl-C
    valPolsador=GPIO.input(11)
    if valPolsador==0:
        continue
    else:
        n=n+1
        if n==10:
            n=0
        print(n)
        display(n,13,15,16,18,19,21,22)
        time.sleep(0.3)
```



7.- Exercicis.

1.- Dissenya el circuit i el programa per controlar un seqüenciador de 6 leds, formant diversos efectes lluminosos.

2.- Dissenya el circuit i el programa per controlar un display de 7 segments, que, en prémer un polsador, s'iniciï un compte enrere des de 9 fins a 0, amb una cadència de 1 s. i, en arribar a 0, s'il·lumini un led. Al cap de 5 s. s'ha d'apagar el led i el display, quedant preparat per tornar a començar el procés en prémer el polsador de nou.